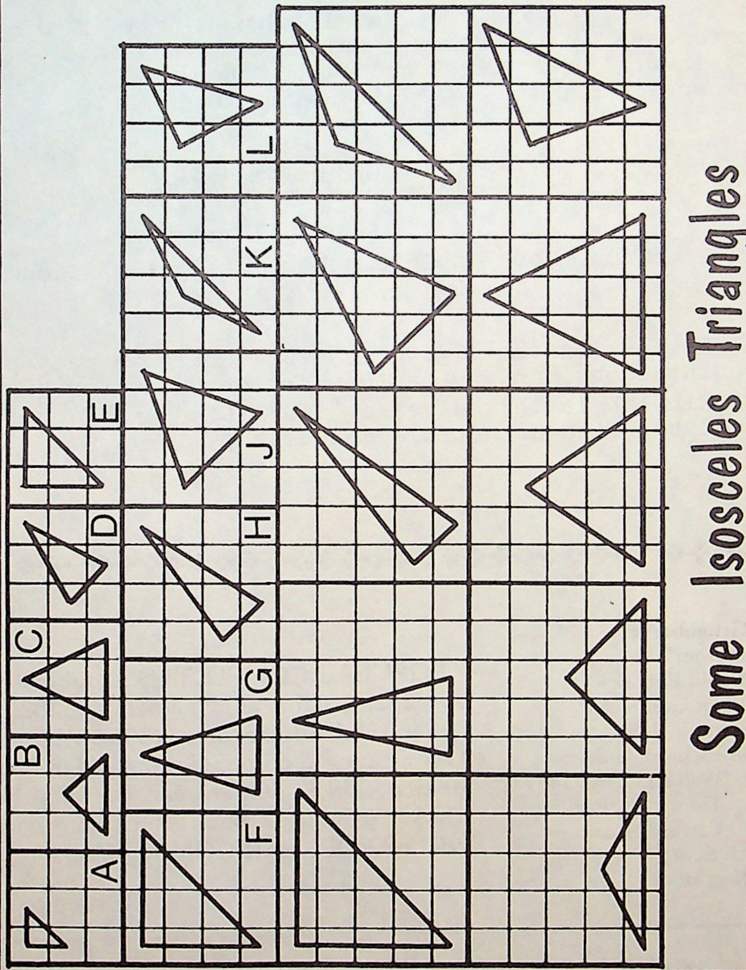
Popular Computing

Volume 9 Number 3

March 1981

96



Isosceles Triangles

In the article "Circle Partitioning" (issue 84), Contributing Editor Thomas R. Parkin showed how a general formula can sometimes be derived from the first few specific cases of a combinatorial problem. If the formula is a polynomial of degree K, then at least K+2 cases must be available.

In that article, Mr. Parkin showed that if the circumference of a circle is marked off into N non-equal parts, then if the points that mark the parts are connected in all possible pairs, the circle is cut up into

$$\frac{N^4 - 2N^3 + 11N^2 + 14N + 24}{24}$$

pieces. Thus, for 7 points, there will be 57 pieces.

[The formula above gives 99 for N = 7, as many people have pointed out, and 99 is the correct value for <u>8</u> points. Mr. Parkin labels this disease "indexitis" and he has promised to write an article about it.]

Publisher: **Audrey Gruenberger**
Editor: **Fred Gruenberger**
Associate Editors: **David Babcock**, **Irwin Greenwald**, **Patrick Hall**
Contributing Editors: **Richard Andree**, **William C. McGee**, **Thomas R. Parkin**, **Edward Ryan**
Art Director: **John G. Scott**
Business Manager: **Ben Moore**

POPULAR COMPUTING is published monthly at Box 272, Calabasas, California 91302. Subscription rate in the United States is $20.50 per year, or $17.50 if remittance accompanies the order. For Canada and Mexico, add $1.50 per year. For all other countries, add $3.50 per year. Back issues $2.50 each.

We used that method in our issue number 89 in deriving the number of squares that exist on an array of squares of size X:

$$\frac{X^4 - X^2}{12}$$

Thus, for the 4 x 4 case (illustrated explicitly in Figure W, page 4, of that issue), there are 20 possible sub-squares on an array of 4 x 4 squares.

This brings us to a new problem:

> On an array of K x K squares, in how many ways can the centers of three of the squares be connected to form an isosceles triangle?

The accompanying Figure shows various possible isosceles triangles on 3 x 3, 4 x 4, and 5 x 5 arrays of squares.

The problem could also be stated in terms of the points on a lattice, but was done in terms of the centers of squares in order to get ready for another problem, similar to FIVESQUARE in issue 94.

We can easily count the number of possible isosceles triangles on a 4 x 4 array by hand:

Type A:	36	Type F:	4
B:	24	G:	8
C:	16	H:	4
D:	16	J:	4
E:	16	K:	4
		L:	16

for a total of 148. The total number of sets of 3 squares that can be selected out of an array of 16 squares is given by:

$$_{16}C_3 = \frac{16 \cdot 15 \cdot 14}{3 \cdot 2} = 560$$

The remaining sets (560 - 148 = 412) either do not form an isosceles triangle, or do not form a triangle at all.

The technique that Mr. Parkin described requires enough cases until, in a pattern of differences, some column exhibits a constant. We can count the first few cases here by hand and form the start of such a difference pattern:

Size of array	Number of isosceles triangles	differences	
2	4		
3	36	32	
4	148	112	80

...but now things get sticky. The 5 x 5 case is not so easy to count by hand, and the cases after that are nearly impossible.

So we turn to the computer, of course. We can write a program to search all sets of 3 on an array of size K x K and tally those that form an isosceles triangle. This program must function in <u>integer</u> arithmetic--floating arithmetic will probably not work.

Given the row and column coordinates of 3 selected squares:

A(I), B(I)

A(J), B(J)

A(K), B(K)

we can form the <u>squares</u> of the three side lengths:

$$X = (A(I) - A(J))^2 + (B(I) - B(J))^2$$

$$Y = (A(I) - A(K))^2 + (B(I) - B(K))^2$$

$$Z = (A(J) - A(K))^2 + (B(J) - B(K))^2$$

Then if X = Y, <u>or</u> X = Z, <u>or</u> Y = Z (it is not possible to have X = Y = Z), the triangle has two equal sides. The values of I, J, and K should run systematically through all possibilities. Thus, the search can be initialized to:

I = 1

J = 2

K = 3

and the logic of Flowchart V will do the job.

Two arrays, A and B, can be initialized to contain the row and column numbers of the array being searched. The following segment of a BASIC program will do this task:

```
200  K = 1
210  FØR I = 1 TØ S
220  FØR J = 1 TØ S
230  B(K) = J
240  A(K) = I
250  K = K+1
260  NEXT J
270  NEXT I
```

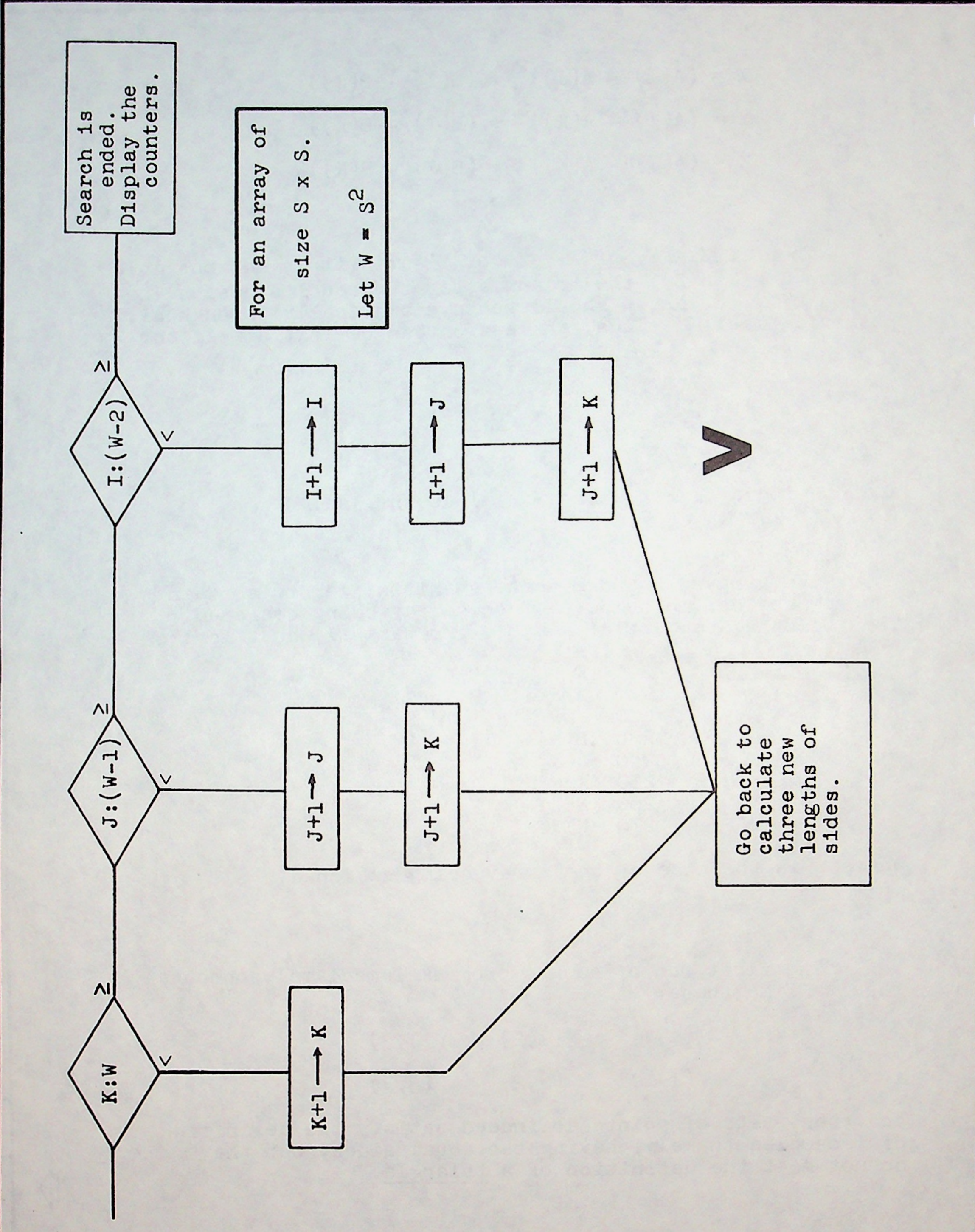
where S is the size of the array being searched.

The first run of such a program immediately outputs results like these:

(1,1)   (1,2)   (1,3)

(1,1)   (2,2)   (3,3)

and these sets of points do indeed satisfy the definition of isosceles (namely, having two equal sides) but they do not meet the definition of a <u>triangle</u>.


K:W
<
≥
K+1 ⟶ K
J:(W-1)
<
≥
J+1 ⟶ J
J+1 ⟶ K
I:(W-2)
<
≥
I+1 ⟶ I
I+1 ⟶ J
J+1 ⟶ K
Search is ended. Display the counters.
For an array of size S x S. Let W = S²
Go back to calculate three new lengths of sides.
V

2	4			
3	36	32		
4	148	112	80	
5	444	296	184	104
6	1064	620	324	140
7	2200	1136	516	192
8	4024	1824	688	172
9	6976	2952	1128	440
10	11284	4308	1356	228
11	17396	6112	1804	448
12	25621	8225	2113	309
13	36812	11191	2966	853

third differences

second differences

first differences

number of triangles

size of the array

Some results of an empirical study of the Isosceles Triangle problem. The accuracy of the results is not guaranteed. The third differences suggest that much more must be done before this problem can be considered as conquered.

A test must be inserted for non-zero area. For three points, the value of the determinant:

$$\begin{vmatrix} A(I) & B(I) & 1 \\ A(J) & B(J) & 1 \\ A(K) & B(K) & 1 \end{vmatrix}$$

provides a measure of the area and, if zero, signals the need to reject the current set of points. The determinant reduces to this:

```
T = A(I)*B(J) + A(J)*B(K) + A(K)*B(I)
   -A(I)*B(K) - A(J)*B(I) - A(K)*B(J)
```

and if T = 0, the set of three points should be rejected and another set examined.

Final results for the first few cases are shown in Table W. The difference pattern does not show any column that is constant.

So the problem is left open: What formula gives the number of isosceles triangles on an array of size S?

PROBLEM **290**

BOOK REVIEW

Coding and Information Theory

by Richard W. Hamming

Prentice-Hall, 1980, $21.95.

Richard Hamming, a pioneer in coding theory, has given us in this book a compact treatment of two subjects traditionally covered separately:

- coding theory: how should symbols of a source alphabet be coded to minimize code lengths and permit error detection and correction?

- information theory: how should information be measured, and what are the upper limits on the amount of information that can be transmitted over a noisy channel?

The two theories are connected by their common concern for errors in the signalling process. As Professor Hamming points out, this signalling can occur "from here to there," as in conventional communication systems, as well as "from now to then" in computer data storage and retrieval.

Professor Hamming originated the concept of "distance" in n-dimensional space as a criterion for the detectability and correctability of errors. Thus, a code with a Hamming distance of at least 2 is required for single error detection, at least 3 for single error correction, and so forth. An informal understanding of this concept is useful in many common coding situations; e.g., names of variables in computer programs should be sufficiently "far apart" that a single keystroke error does not inadvertently convert a reference to one variable into a reference to another.

The book assumes no mathematical skills beyond simple calculus and a little probability theory. Everything else is developed as it is needed. This is quite a bit, as it turns out, including Bayes' theorem, the Gamma function, Stirling's approximation, the law of large numbers, and n-dimensional Euclidean space inequalities. The style is informal, but not at the expense of a careful and correct exposition. The clarity of the exposition ranges from somewhat rocky at places in the early sections on coding theory, to very smooth in the later sections on information theory.

As a simple example of error detection, Professor Hamming explains the International Standard Book Number (ISBN), a 10-digit number assigned by publishers to their books, in which the last digit is a check digit. Hamming will be relieved to learn that the ISBN assigned by Prentice-Hall to his book, 0-13-139139-9, is correct.

--reviewed by W. C. McGee

PROBLEM 291

In the book Structured BASIC and Beyond, by Wayne Amsbury, exercise PG2.4 is the following:

> Consider a sequence of numbers which has the property that beginning with the 4th one, each number is the sum of the previous one and twice the one before that minus the one before that. For example, the 8th one is (the 7th one) plus twice (the 6th one) minus (the 5th one). Suppose that the first three numbers in the sequence are 1, 2, and 3. Write a program that finds the 15th one.

In other words:

$$T_{i+1} = T_i + 2(T_{i-1}) - T_{i-2}.$$

(This is not a computer problem, as it is stated. It can be done by hand, or with a pocket calculator, in about the time it would take to find a coding sheet and draw a flowchart. The problem is an exercise in rudimentary coding, and not too good at that. Why stop at the 15th term? How is the program to be tested?)

The sequence does suggest some interesting problems, however, that may be computer problems:

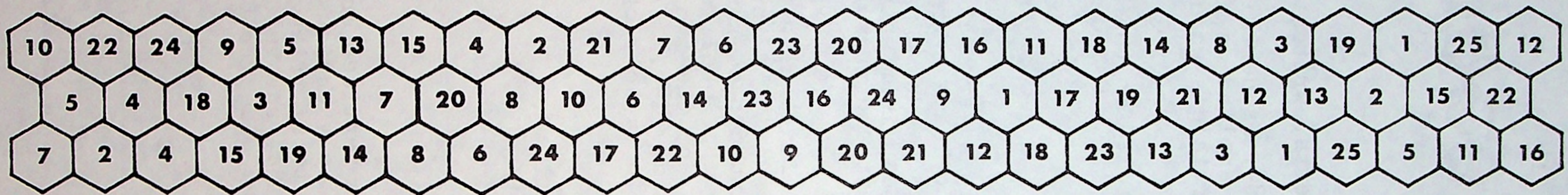
1. The ratio of two successive terms takes on values like these:

1.848...
1.770...
1.824...
1.787...
1.813...

which suggests that it may converge. To what?

2. Starting values of 0,0,0 will present a sequence that converges immediately. Will any other set of starting values lead to a converging sequence?

The La Jolla Scramble



In the honeycomb pattern shown here, successive rows of hexagons have 25 or 24 cells. In each row, the numbers from 1 to 24 or 1 to 25 are inserted in random order. For each such row that is added to the pattern, how many pairs of connecting cells between the rows will contain the same number, or have two numbers differing by one?

For the first two rows in the illustration, we have the following pairs: 7-6, 23-23, 18-17, 18-19, and 1-2.

Between the second and third rows, we have: 4-4, 7-8, and 17-18.

Thus, through two stages, the process seems to average 4 per row.

The Problem is: what will be the average count for thousands of such rows?

PROBLEM **292**

feedback

Michael Beeler, Belmont, Massachusetts, writes:

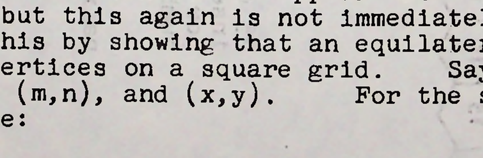
NOSQUARE, the case study in issue 89, is certainly an interesting problem and imaginatively applied as a class exercise. Two implications of the article's wording bother me, however.

One is that the arrival at the formula for the number of squares on an X by X grid:

$$Y = \frac{X^4 - X^2}{12}$$

is necessarily correct. I agree that it's very likely, but the discussion in the article seems to leave open the possibility of strange behavior on grids above some threshold size. A derivation based on explicit enumeration is slightly messy but finds the same result and seems more satisfying.

Second is the assertion that a quadrilateral with distinct vertices on a square grid with exactly 7 distance pairs equal must be a square. This also can be shown rigorously. The seven distance pairs being equal forces four of the six distances to be the same, and the two remaining distances to be equal to a second value. This makes the figure a square, a sort of "Y," or a sort of "kite."



Neither the Y nor the kite can appear with vertices on a square grid, but this again is not immediately obvious. We can show this by showing that an equilateral triangle cannot have vertices on a square grid. Say the vertices are at (0,0), (m,n), and (x,y). For the sides to be equal, we have:

$$x^2 + y^2 = s^2$$

$$m^2 + n^2 = s^2$$

$$(m-x)^2 + (y-n)^2 = s^2$$

Solving for x and y, we get:

$$x = \frac{m \pm n\sqrt{3}}{2}$$

$$y = \frac{n \mp m\sqrt{3}}{2}$$

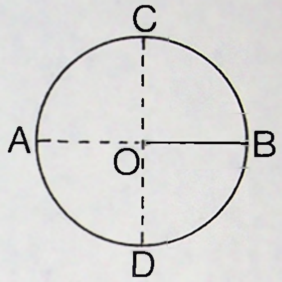
Obviously, if m and n are integers, neither x nor y are. Since both the Y and the kite contain equilateral triangles, they can't appear with vertices on the square grid. The only case remaining is the square, so the article's assertion holds.

What bothers me about the way these appeared in the article, though, is an "it seems to work, so use it" attitude; I hope that students are told that there is rigorous support for the algorithms.

At the risk of being pedantic, I should mention two speed improvements. First, instead of throwing rejected points back into the pool, mark them as already considered and rejected. Select randomly from a pool of only those never considered.

And second, a squareness determination can be performed more quickly as follows. Take the candidate point and separately pair it with each point in the "chosen" pool. For each such pair, the pair may be the side of only two distinct squares, or the diagonal of only one square. For each of these three squares, compute (easy) where the other two points would lie. See whether both are in the chosen pool (an off-the-grid check may be useful here).

Problem Solution



The following is from Dr. Rudi Borth, DON MILLS, Ontario, Canada.

No answer to Problem 129 in issue 38 ("A Circuitous Race") seems to have been published, so I am sending you mine.

Apart from the usual challenge to understand the underlying logic and to find a simple way of coding that works, the problem turned out to be exceptional in that it required many program steps more than any other POPULAR COMPUTING problem I have so far wrestled with.

[The Problem was this: Given a circular road of length 100 miles. A car starts at O, goes to B, and travels the circle counterclockwise, back to B. It then crosses the circle from B to A, after which it again circles the road. This procedure (a complete circle, followed by a crossing on the diameter) is repeated for twenty complete revolutions, after which the car returns to the center. There are four such cars. No. 2 starts at O and heads North; No. 3 starts West; No. 4 starts South; the four cars all start at the same time, and each makes a trip the same way. Their speeds are 67, 29, 59, and 41 miles per hour respectively. How many times has any car overtaken and passed another car?]

The race turns out to be a lonely one. Despite the length of time taken by this endurance test (39.35 to 90.92 hours, depending on the speed), any two cars meet only a total of 7 times.

Car 2 is overtaken by car 1 in section CA, and car 4 by car 3 in section BC. Both events occur during round 1, and there are no other overtakings. While these could have happened both in the circular and the radial sections, passing is limited to pairs of cars moving in opposite directions on a diameter: either cars 1 and 3 on AOB, or cars 2 and 4 on COD.

Cars 1 and 3 pass each other only once; namely, on OA during round 1. Cars 2 and 4 pass each other twice in section OC, during their rounds 5, 7, and 15, 20, respectively; and twice in section OD, during rounds 9, 13, and 10, 15, respectively.

And that's it!

These revelations were produced by the following method.

1. From the car speeds (v) given, calculate the times it takes to drive through a circular section (25/v for a quarter circle) or a radial section ($50/\pi v$ for a half diameter).

2. Number the race track sections in some logical way related to the problem:

OD	OA	CO	BO	OB	OC	AO	DO	BC	CA	AD	DB
-4	-3	-2	-1	1	2	3	4	5	6	7	8

using $\pm n$ to label opposite directions of a radius helps in coding.

3. The conditions of the problem can be reduced to the statement that each car runs 10 times over the same route. For car 1, the route is OBCADBOADBCAO, and it corresponds to the 12-section sequence 1 5678 -1 -3 7856 3; for car 2, the sequence is 2 6785 -2 -4 8567 4; and so on, similarly for cars 3 and 4.

4. The race begins at time 0. Follow each car through its 10 route replicates, and calculate and store the times when it enters and leaves each section. Strictly speaking, only the exit times are necessary (an entry time equals, of course, the time of exit from the preceding section), but some waste of array space makes the fact that each point ABCDO is a fork easier to handle.

5. For a car to overtake another in any section, it must enter the section after, and in the same direction as, the other, and leave it before the other. Let the entry times of the two cars be t_1 and t_1' and the exit times be t_2 and t_2'. A car has been overtaken if the differences $(t_1 - t_1')$ and $(t_2 - t_2')$ have opposite signs; that is, their product is negative:

$$(t_1 - t_1')\,(t_2 - t_2') < 0$$

6. For two cars to pass in any section, they must enter it in opposite directions before the other has left it; that is, the differences $t_1 - t_2'$ and $t_2 - t_1'$ have opposite signs; their product is negative:

$$(t_1 - t_2')\,(t_2 - t_1') < 0.$$

7. Apply these two criteria to all possible and appropriate combinations of two cars and a section, omitting those which the conditions of the problem preclude from occurring. Count and document the events detected.

A program implementing steps outlined above, 2848 bytes long and requiring about 17600 bytes of array space, took 1 minute, 55 seconds to run on an H-P 9825.

In view of the actual numbers given, the possibility that cars might meet exactly at points ABCDO was ignored. As expected, a refined program taking this possibility into account detected no such events.

In the absence of any bright ideas, the program was tested only superficially and may be liable to demolition. Testing included manual spot checks of the table generated in step 4 (and printed for this purpose), a check of the total time and distance traveled by each car, and some varying of car speeds--all with plausible or expected results.

Problem 283 (Fivesquare) in issue 94 gave a scoring scheme for sub-squares on a 5 x 5 array.

For four squares selected from a 5 x 5 array of squares (the centers of those squares forming a sub-square), points were allocated as follows:

4 points if the four numbers are prime
1 point if they are all composite
1 point if they have a common factor
10 points if they form an arithmetic progression.

The numbers in the 25 cells are to be some permutation of the numbers from 1 to 25. The 50 possible ways of forming the sub-squares were given on the cover of issue 94.

Charles Haspel, of New York City, produced this arrangement in the array:

1	2	7	8	9
3	4	12	10	11
6	5	13	21	20
15	16	14	22	23
17	18	19	24	25

by hand (using a computer program only to calculate the score). Mr. Haspel comments: "The arrangement is very symmetrical, and in fact was produced by trying to take advantage of symmetries. I have no idea if it is optimal."

(In a later communication, Mr. Haspel found an even better result, having a count of 203:

1	2	17	15	13
3	4	18	16	14
5	6	25	24	23
8	10	12	22	21
7	9	11	20	19

using an APL program that generated and scored 18 permutations of a given hand-generated solution.)

Problem Solution

Problem Solution

FOURWAY was one of the first original computing problems presented in POPULAR COMPUTING, in issue number 3. It can best be explained in terms of the 3 x 3 case.

Given an array:

1	1	1
1	1	1
1	1	1

of nine cells, each containing the number one to start. Each game begins at the center square and moves are made according to this plan:

If the cell contains:	Replace it with:	And move to the cell to the:
1	2	north
2	3	east
3	4	south
4	1	west

until a move is made that goes off the array.

Many problems can thus be posed; for example, how frequently does a game exit from each of the 12 possible exit positions? The most difficult problem, however, is this: how many games will it take to restore the pattern of number to all ones? In the 3 x 3 case, the answer is 16, which can be readily worked out by hand. The contents of the cells at the end of each of the 16 games is as follows:

```
1 2 1     1 2 2     2 2 2     2 3 3
1 2 1     1 3 2     2 1 2     2 2 2
1 1 1     1 1 1     1 2 1     1 2 1

2 3 3     2 3 3     2 4 3     2 4 3
2 3 3     2 4 4     3 4 1     1 1 1
1 2 1     1 3 3     1 4 3     2 4 3

3 2 3     3 2 4     4 4 1     4 4 1
1 2 1     1 3 3     3 3 4     3 4 4
2 4 3     2 4 3     3 2 4     3 4 1

4 4 1     1 1 1     1 1 1     1 1 1
4 1 4     4 2 4     4 4 1     1 1 1
4 4 1     4 4 1     1 1 1     1 1 1
```

In 1973, cases 3, 5, 7, and 9 had been analyzed by computer. This research has now been extended to the next case, and the current status is:

Case	Number of games to return to the original pattern:
3	16
5	104
7	544
9	146 248
11	7 889 840

FOURWAY was designed to demonstrate two characteristics of computing that make it different from previous tools:

1. There are simple, well-defined procedures that cannot be carried out by any other means than the computer; and

2. The outcome of the computer program is frequently unpredictable, especially by the person who wrote the program.

FOURWAY seems to epitomize both these characteristics.

Plotting the above data indicates that the 13 x 13 case may require nearly a billion games to return to the original pattern. The 11 x 11 result consumed 37 hours of CPU time, in 6502 machine language.

PROBLEM **293**

The 196 problem was discussed in our issue number 30. It started with the Palindrome problem: Take any positive integer, reverse its digits and add; repeat this process until the sum is a palindrome. Most numbers reach the final status in a few stages. The number 196 represents the group of numbers that apparently will never evolve into a palindrome. The first 20 stages of the process for 196 are shown here.

The 196 starting value grows to an eleven-digit number in 20 stages; the number of stages-per-digit at that point is thus 1.818. Extensive graphs of the ratio (stages per digit) were shown in issue 30; at times the ratio seems to be quite erratic.

But this is still another problem that naturally lends itself to decimal-digit-per-byte processing. The ratio cited above <u>does</u> stabilize somewhat, as given here:

Range of stage numbers	Ratio
14800 -- 16400	2.414
21000 -- 24000	2.410
31000 -- 35000	2.408
39000 -- 44000	2.404
50000 -- 52000	2.406

This data is only a tiny sample of an infinite process. What can be concluded about this ratio as the number of stages is extended indefinitely?

	196	691
1	887	788
2	1675	5761
3	7436	6347
4	13783	38731
5	52514	41525
6	94039	93049
7	187088	880781
8	1067869	9687601
9	10755470	07455701
10	18211171	17111281
11	35322452	25422353
12	60744805	50844706
13	111589511	115985111
14	227574622	226475722
15	454050344	443050454
16	897100798	897001798
17	1794102596	6952014971
18	8746117567	7657116478
19	16403234045	54043230461
20	70446464506	